Technical Report

Mobile health checking is a huge market nowadays, approaching a number of $300 billions. Used to help with diagnosis, information about different medication or with measurements of lab results, these applications can be split in two main categories: intended for patients or intended for doctors and other medical staff. Because our project is part of the latter category we will study the market to see what are the capabilities of the applications available currently.

Epocrates is one such application, used by professional medical staff to quickly get information about medicaments and patient measurements, while also having a feature that allows diagnosis for some given symptoms, yet, some of that functionality is really expensive, worse is that there is no restriction when it comes to users, anyone can download and use the application, which may lead to problems. UpToDate is a product that solves this issue. While it has free content for normal users, UpToDate restricts it's more relevant functionalities, only certified practitioners can access them.

Our project implies two main phases. The collection of existing medical records and the diagnosis of new incomplete records. For the first phase, a doctor would complete an in depth form with relevant data of past cases. The fields of the form are based on the existing categories of real medical records, while loose description will be written in a separate field which will undergo further processing. This processing consists of applying different machine learning algorithms in order to separate a large body of text in a few key attributes. This phase can be evaluated by the numbers of records that we have in storage and by the fidelity of the correlations between our data and the real data added by the doctors.

Only after we have a respectable amount of records in our database we should be able to use the second phase functionality effectively. This phase implies that the user fills up another incomplete form with measurements and symptoms. The application will then use a clustering algorithm to categorise the input with some already existing records, thus giving an exact diagnosis. There are a multitude of highly efficient clustering algorithm including:

* K-means algorithm. One of the basic clustering algorithms, used to categorise n entities to k centers of clusters. In our case these centers would all be different diagnostics. It's main drawback is that it works with a constant number of clusters (k).
* ID3. A machine learning algorithm used to generate a decision tree from a dataset.
* Agglomerative Hierarchical Clustering. Is the most common type of hierarchical clustering used to group entities in clusters based on their similarity.

Technical Solution

As a solution for our data storage we will proceed with a MySQL based database, we chose to do this because of a few main reasons:

* **Transaction based queries**
* Given that MySQL transactions don’t commit in case an error would appear during it’s runtime, they provide a very secure way to store data, given that it can be considered a second layer of tries and catches that don’t affect the integrity of the data
* **High Availability**
* MySQL is designed to process millions of queries and thousands of transactions while ensuring unique memory caches, full-text indexes and optimum speed
* **Scalability**
* In order to have high availability for all our customers MySQL can be an easy to scale tool, that can be accessed using cloud methods, AWS offers some ways to scale the number of database masters and slaves that are working at a given time
* **Reliability**
* With the recent changes in how user data should be handled and the addition of the GDPR, MySQL ensures data security with its data protection features, data encryption prevents unauthorized viewing of data and SSH and SSL support ensure safer connections
* It also features a powerful mechanism that restricts server access to authorized users and has the ability to block users even at the man-machine level
* the data backup feature facilitates point-in-time recovery
* **Quick-Start**
* You can go from software download to complete installation in just 15 minutes
* very quick, regardless of the underlying platform
* **Relational engines**
* The InnoDB engine constitutes a very powerful tool, given that we want to check the identities of the persons adding our data, assign regions for the inserted observations; foreign key constraints to maintain data integrity; fine grained locking-mechanism

In order for the easy and efficient development of our application, having in mind that it needs to be accessible from both desktops and mobile stores, we decided to use Ionic. Ionic Framework is a free, open source mobile UI toolkit for developing high-quality cross-platform apps for native iOS, Android, and the web, all from a single codebase. Some of Ionics greatest advantages are:

* Free & Open Source

Ionic Framework is a 100% free and open source project, licensed under MIT. It will always remain free to use, powered by a massive world-wide community.

* Fully Cross-Platform

Build progressive web and native mobile apps for every major app store, with one codebase. Ionic works and looks beautiful wherever it runs.

* Premier Native Plugins

Use over 120 native device features like Bluetooth, HealthKit, Finger Print Auth, and more with Cordova/PhoneGap plugins and TypeScript extensions.

* First-class Documentation

Built with real app examples, component demos, guides, and how-to’s to get you up and running with mobile apps faster than ever before.

Our solution

Our solution implies two big steps: keyword collection and clustering of the records.

Keyword collection:

The collection of important tokens found in the text of the medical records that we receive from the client application. These are stored into the database, linked to the medical record entity for easy access. Each keyword entity has a name, a category and a significance coefficient. The category is the field in which said keyword was found (nutrition, general health…), and the significance coefficient is the amount of importance that is held by the keyword in the phrase context.

In other to gather keywords we used two external API’s used to find such words, given an English text, the API’s would return a list of keywords. The first problem is that there is no keyword collector made to work with the Romanian language. So we must start by translating the text from Romanian to the English language. We did this by using the Watson Platform translator. Once we got a hold of the translation we can begin the keyword collection.

We call both API’s asynchronously, and wait for both of them to finish executing. After we get the results we process them in order to gather the keywords. Each keyword is then lemmatized in order to be as general as possible, after that the duplicates are eliminated and the results are added into the database for further processing.

Clustering medical records:

In order to classify different medical records to a given diagnosis we must cluster them with the rest of the medical records that we have stored in our database. In order to do this we need a similarity function that will return a number, the bigger the number the further away are two medical records from one another. The maximum score is 20, one point for each category. As the smaller the score the better, if two categories have similar keywords the keywords will be subtracted by 1 and so on. Numeric values like height, weight or waist have more potency in computing the fitness.

This process will result in a hierarchy of clusters, from general to specific. Out of the levels of the hierarchy we want to choose the right one that makes a good balance between the number of examples and the percentage of the true diagnosis. Thus the right cluster will have a respectable amount of items with a good percentage of ‘understanding’ between each other.

Future Work.

There is a lot of untapped potential for this project. Starting with the front-end part, the application does not have user authentication and, even though it’s mobile responsive, we lack a true mobile application (android or IOS). The application could also do with a more unique theme. Multiple roles should also be implemented, each with unique functions.

On the back-end part the application should receive some further performance improvements. Another good feature would be the implementation of different clustering algorithms, this will enable testing and comparing the different methods in order to find the best one. The back-end part should also expose a draft based file system, this would allow for the storing of medical records as incomplete drafts. User should be able to complete a medical record at a later date or store it indefinitely.

The back-end should also get an authorization/authentication treatment just like the front-end, exposing routes and functionalities only to users that have a required role. The database should be moved in the cloud and so should the REST API and the client server. We should have three environments: development, quality assurance and production, each one with their own context and database.

Results

The query endpoint role is to gather and filter the medical records stored into the database. It works well and it’s quite reliable, as there are no failures, even though the speed is not remarkable, as it’s bottlenecked by the network transfer speed.

The translate endpoint is used to translate pieces of text from Romanian to English. The translation is done via an external API that, when overcharged, will take longer than anticipated or even fail completely. Either way, on smaller samples the endpoint will do fine.

The keyword endpoint is used to gather all the keywords from a piece of text. The processing is done via two external API’s whose results are then combined in order to give a better and more precise response. Every keyword is lemmatized using another API, and duplicates are eliminated. There has been performance issues with this endpoint, so we put a lot of work in order to fix them. This is clearly shown by watching the time performance displayed in the chart. Every action is done in parallel and every call was tweaked to improve performance. When overcharged the endpoint can have failures as we use many external API’s.

The clustering endpoint receives a medical record and returns a diagnosis based on the clustering of said medical record with the records that we have in the database. Everything is done with the help of a library, so no external calls, and the fitness function was tweaked to improve performance. This can be seen by examining the chart. As we don’t call any external API’s there are no failures when overcharged.

Comparison of methods

We tested two API’s used to gather keywords Algorithmia and ParallelDots. We have found that Algorithmia is faster, easier to use and even comes up with more keywords, yet ParallelDots API had some strong points too. Although it ran a bit slower and could fail by not detecting some keywords, ParallelDots also gave information about the keywords significance in the phrase. This is very important information that really helps us in clustering the medical reports, so we decided to use both of them, running them concurrently and then combining their results.

Links and other materials

* University of Bonn and the Charité - have shown that artificial intelligence can be used to diagnose rare diseases more efficiently and reliably.
* Klaus Kayser, JĂźrgen GĂśrtler, Milica Bogovac, Aleksandar Bogovac, Torsten Goldmann, Ekkehard Vollmer, Gian Kayser - AI used in image analysis in histopathology.
* Jiang M, Huang Y, Fan JW, Tang B, Denny J, Xu H - From School of Biomedical Informatics, The University of Texas Health Science Center at Houston, Research in parsing, analise, and clasify medical text and data.
* Lale Özyilmaz - From Yildiz Technical University, Istanbul, Turkey, uses neuronal networks in research of Parkinson’s Disease.
* NaCTeM is operated by the University of Manchester - is the first publicly-funded text mining centre in the world, used for research in medical text mineing.
* Diagnosis of Clostridium difficile-associated disease: examination of multiple algorithms using toxin EIA, glutamate dehydrogenase EIA and loop-mediated isothermal amplification.  
  ([https://search.proquest.com/openview/cf78fe8565023013b0150db59e4b8821/1?pq-origsite=gscholar HYPERLINK](https://search.proquest.com/openview/cf78fe8565023013b0150db59e4b8821/1?pq-origsite=gscholar&cbl=4969) )
* Artificial Intelligence in Personalized Medicine Application of AI Algorithms in Solving Personalized Medicine Problems.

(<https://www.researchgate.net/profile/Jamilu_Awwalu/publication/282624363_Artificial_Intelligence_in_Personalized_Medicine_Application_of_AI_Algorithms_in_Solving_Personalized_Medicine_Problems/links/5958ac5b458515ea4c62af76/Artificial-Intelligence-in-Personalized-Medicine-Application-of-AI-Algorithms-in-Solving-Personalized-Medicine-Problems.pdf>)

* Diagnosis of thyroid disease using artificial neural network methods.

(<https://ieeexplore.ieee.org/abstract/document/1199031>)

* Pivotal trial of an autonomous AI-based diagnostic system for detection of diabetic retinopathy in primary care offices.

(<https://www.nature.com/articles/s41746-018-0040-6/>)

* Proposed diagnostic criteria and nosology of acute transverse myelitis.

(<https://miami.pure.elsevier.com/en/publications/proposed-diagnostic-criteria-and-nosology-of-acute-transverse-mye>)

* Artificial neural networks for diagnosis of hepatitis disease.

(<https://ieeexplore.ieee.org/abstract/document/1223422>)

* AI (artificial intelligence) in histopathology--from image analysis to automated diagnosis.

(<https://journals.viamedica.pl/folia_histochemica_cytobiologica/article/view/4316>)

* How artificial intelligence can help detect rare diseases.

(<https://www.sciencedaily.com/releases/2019/06/190606133805.htm>)